

On the Complexity of Metric Dimension

Josep Diaz¹, Olli Pottonen¹, Maria Serna¹, and Erik Jan van Leeuwen²

¹ Departament de Llenguatges i Sistemes Informatics, UPC, Barcelona, Spain.
 diaz,mjserna@lsi.upc.edu, olli.pottonen@iki.fi

² Dept. Computer and System Sciences, University of Rome “La Sapienza”, Italy
 E.J.van.Leeuwen@ii.uib.no

Abstract. The metric dimension of a graph G is the size of a smallest subset $L \subseteq V(G)$ such that for any $x, y \in V(G)$ there is a $z \in L$ such that the graph distance between x and z differs from the graph distance between y and z . Even though this notion has been part of the literature for almost 40 years, the computational complexity of determining the metric dimension of a graph is still very unclear. Essentially, we only know the problem to be NP-hard for general graphs, to be polynomial-time solvable on trees, and to have a $\log n$ -approximation algorithm for general graphs. In this paper, we show tight complexity boundaries for the METRIC DIMENSION problem. We achieve this by giving two complementary results. First, we show that the METRIC DIMENSION problem on bounded-degree planar graphs is NP-complete. Then, we give a polynomial-time algorithm for determining the metric dimension of outerplanar graphs.

1 Introduction

Given a graph $G = (V, E)$, its *metric dimension* is the cardinality of a smallest set $L \subseteq V$ such that for every pair $x, y \in V$, there is a $z \in L$ such that the length of a shortest path from z to x is different from the length of a shortest path from z to y . In this case we say that x, y are *resolved* by z and L . Elements of the set L are called *landmarks*. A set $L \subseteq V$ that resolves all pairs of vertices is called a *resolving set*. The problem of finding the metric dimension of a given graph G is called METRIC DIMENSION, but is also known as *Harary’s problem*, and the *locating number* or *rigidity* problem. The problem was defined independently by Harary and Melter [12] and Slater [18], and has received a lot of attention from researchers in different disciplines (see e.g. [2,13] and references therein). For example, a recent survey by Bailey and Cameron [2] notes an interesting connection to group theory and graph isomorphism.

Computing the metric dimension has proved to be a hard problem, and there are very few results on the computational complexity of the problem. The decision version of METRIC DIMENSION is known to be NP-complete on general graphs [11,15,16]. Khuller et al. [16] gave a linear-time algorithm to compute the metric dimension on a tree. Characterizations for graphs with metric dimension 1 and 2 are also given in the same work. Moreover, [16] also provides a $2 \log n$ -approximation algorithm for the metric dimension of any graph. Beerliova et al. [3] proved that the metric dimension of a general graph cannot be

approximated within $o(\ln n)$, unless $P=NP$. Hauptmann et al. [13] improved on this by giving a $(1 - \epsilon) \log n$ inapproximability result under mild complexity assumptions. They also show that determining the metric dimension is APX-hard for bounded-degree graphs, and APX-complete for superdense graphs (graphs for which the complement is a sparse graph). To the best of our knowledge, no further results are known about the complexity of this problem.

Our Results In the present work, we narrow the tractability gap of METRIC DIMENSION. From the hardness side, we show that METRIC DIMENSION on planar graphs, called PLANAR METRIC DIMENSION, is NP-hard, even for bounded degree planar graphs. Even though Garey and Johnson considered METRIC DIMENSION over 30 years ago in their book [11, Problem GT61], but the complexity of the planar version of the problem remained unknown thus far. From the algorithmic side, we show that there is a polynomial-time algorithm to find the metric dimension of outerplanar graphs. This significantly improves the 15-year-old algorithm on trees by Khuller et al. [16].

Our algorithm for outerplanar graphs presents a leap forward in the type of problems that can be solved on such graphs. Many graph problems, e.g. independent set, are *local* in the sense that a vertex has direct effect on its neighbors only—indirectly the effect propagates farther. The main obstacle in the METRIC DIMENSION problem is that it is *nonlocal*: a landmark can resolve faraway vertices. This means that standard approaches fail to work for it. In her seminal paper, Baker writes that the dynamic programming technique therein “*will work for problems that involve local conditions on nodes and edges*” [4]. This does not include METRIC DIMENSION. For the same reason, the generalizations of Baker’s work by Eppstein [10] and Courcelle [6] do not apply either, and a different kind of approach is necessary.

The crux then to both our hardness and algorithmic results is our ability to constrain the effects of a landmark to a small area. The NP-hardness proof constructs a specific family of planar graphs for which METRIC DIMENSION is essentially a local problem. The algorithm uses a tree structure to traverse the graph, together with several data structures that track the influence of landmarks on other vertices. The structures that we employ differ significantly from the ones used in standard bounded-treewidth techniques. Moreover, whereas outerplanar graphs have constant treewidth [5], the tree structure used in our approach leads to a decomposition that can have arbitrary width.

We believe that our algorithmic techniques present a generalization of Baker’s work, and are of independent interest. In particular, they might lead to (new) algorithms for a broad class of non-local problems (see also Section 4).

Overview of the NP-Hardness Proof As a corollary of the work by Dahlhaus et al. [8], we prove a new version of PLANAR 3-SAT to be NP-complete. We reduce this problem to METRIC DIMENSION. This is done by constructing a planar graph consisting of clause gadgets and variable gadgets. Let n be the number of variables. Each variable gadget must have 4 landmarks, 3 at known, specific locations, but for the fourth we have three different choices. They correspond

to the variable being true, false, or undefined. These $4n$ landmarks are a metric base if and only if they resolve all pairs in clause gadgets, which happens only if they correspond to a satisfying truth assignment of the SAT-instance.

Overview of the Algorithm First, we characterize resolving sets in outerplanar graphs by giving two necessary and sufficient requirements for an arbitrary vertex set to be a resolving set. Then, taking as a base the dual of the biconnected components of the graph G , we define a tree T . Vertices of T corresponds to faces and cut vertices of G , and edges to inner edges and bridges of G . The algorithm uses dynamic programming to process T , starting at the leaves and advancing towards the root. Data structures called *boundary conditions* and *configurations* are used on edges and vertices of T , respectively, to describe how the landmarks are positioned.

Boundary conditions track the effects of landmarks placed in the already processed part of the graph and the possible effects of sets of landmarks to be placed in the unexplored paths of the graphs.

The main algorithmic novelty lies in the configurations, which control the process of combining the boundary conditions on edges towards children of the current vertex $v' \in V(T)$ into a boundary condition on the edge towards the parent of v' . The configurations highly depend on the vertices of G represented by v' . However, even though the number of vertices of G represented by v' might be unbounded, we can show that the total number of relevant configurations is only polynomial. The use of configurations also presents a stark contrast with the techniques used in bounded treewidth algorithms, where the combination process commonly is a simple static procedure.

Preliminaries For basic notions and results in graph theory, we refer the reader to any textbook on the topic, e.g. [9]. All graphs are finite, undirected, and unless otherwise stated, connected. We use the notation (u, v) to denote an edge from u to v . By distance $d(u, v)$ we mean the graph distance. The vertex and edge sets of H are denoted by $V(G)$ and $E(G)$, respectively. Recall that a graph is *planar* if and only if it does not contain a subgraph that is a subdivision of K_5 or $K_{3,3}$, and it is *outerplanar* if and only if it does not contain a subdivision of K_4 or $K_{2,3}$. A graph G has a *cut vertex* if the removal of that vertex disconnects the graph into two components. A graph is a *biconnected* if it has no cut vertices. If G is a biconnected outerplanar graph, G has a planar embedding in which the edges on the boundary of the outer face form a Hamiltonian cycle. We call those edges *outer edges* and the rest of the edges are called *inner edges*. Given G and $v \in V(G)$, $\mathcal{N}(v)$ denotes the set of neighbors of v in G . Given a set S , we denote by $\mathcal{P}(S)$ the power set of S .

2 NP-hardness on planar graphs

It is well known that 3-SAT is NP-complete (see [11]). We require a special planar version of 3-SAT to be NP-complete.

Definition 1 Let Ψ be a boolean formula, which uses the set of variables V and has the set of clauses C . Then the graph $G_\Psi = (V \cup C, E)$, where $E = \{(v, C) \mid v \in V, c \in C, v \in C\}$, is the clause-variable graph of Ψ .

With the notation $v \in C$ we mean that variable v (or its negation) occurs in clause C . Observe that G_Ψ is always bipartite.

Theorem 2 ([8, p. 877]) *The problem of deciding whether a boolean formula Ψ is satisfiable is NP-complete, even if*

- every variable occurs in exactly three clauses (twice positive, once negative),
- every clause contains two or three distinct variables, and
- G_Ψ is planar.

As a corollary of Theorem 2, we get the following result, which is the starting point of our work.

Corollary 3 *The problem of deciding whether a boolean formula Ψ is satisfiable is NP-complete, even if*

- every variable occurs exactly once negatively and once or twice positively,
- every clause contains two or three distinct variables,
- every clause with three distinct variables contains at least one negative literal, and
- G_Ψ is planar.

We call this decision problem 1-NEGATIVE PLANAR 3-SAT.

Proof. Let Ψ be a boolean formula satisfying the constraints of Theorem 2. By modifying Ψ we will construct a formula Ψ' that fulfills all the constraints of the theorem statement and is satisfiable if and only if Ψ is satisfiable.

We only need to eliminate those clauses containing three positive literals. Suppose that $x \vee y \vee z$ is such a clause of Ψ with distinct variables x, y, z . Now add a new variable x' , and replace the original clause by the clauses $x \vee x'$ and $\neg x' \vee y \vee z$. This completes the construction of Ψ' . As this construction replaces some edges of G_Ψ with paths, it preserves planarity.

Given a satisfying truth assignment of Ψ , we get a satisfying assignment of Ψ' by setting $x' = \neg x$. A satisfying assignment of Ψ' implies a satisfying assignment of Ψ . So Ψ' is satisfiable if and only if Ψ is. The theorem now follows straightforwardly from Theorem 2.

To prove that PLANAR METRIC DIMENSION is NP-hard, we will give a reduction from 1-NEGATIVE PLANAR 3-SAT. The idea behind the graph constructed in this reduction is the following. Given an instance Ψ of 1-NEGATIVE PLANAR 3-SAT, we first find a planar embedding of its clause-variable graph G_Ψ . We then replace each variable vertex of G_Ψ by a *variable gadget* (see Figure 1), and each clause vertex of G_Ψ by a *clause gadget* (see Figure 2). By identifying vertices of variable gadgets and vertices of clause gadgets in an appropriate way (see

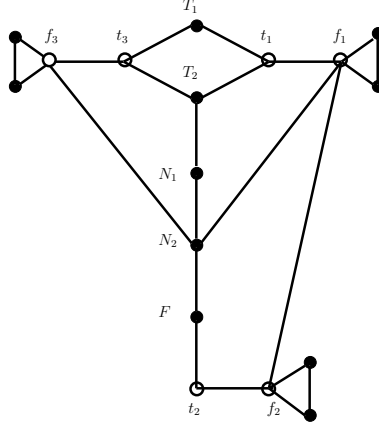


Fig. 1. The variable gadget.

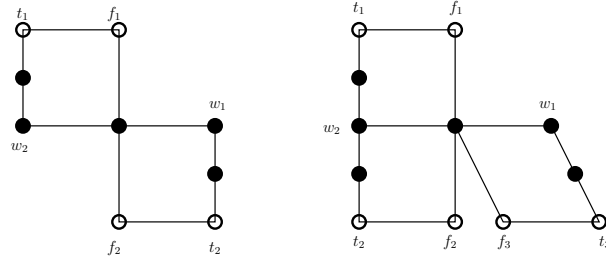


Fig. 2. The clause gadget for a clause with two variables (left), and the gadget for a clause with three variables (right).

Figure 4), we obtain a planar graph H_ψ that will be our instance of PLANAR METRIC DIMENSION.

We now describe our construction in detail. Consider a planar embedding of G_ψ , which can be found in linear time [14]. We first replace each variable vertex of G_ψ by a variable gadget. In Figure 1, the vertices marked by unfilled circles will be identified with vertices from a clause gadget later on. There are three groups (connected components) of such vertices in the figure. The groups containing vertices (t_1, f_1) and (t_3, f_3) will be identified with vertices in clause gadgets where this variable appears positively in the corresponding clause; the group containing (t_2, f_2) will be identified with vertices in clause gadgets where this variable appears negatively. By rotating and contorting the variable gadget appropriately, we can ensure that the three groups point into the right direction (i.e. the negative-appearance group faces the clause vertex where the variable appears negatively).

Next, we replace the clause vertices by clause gadgets. The exact gadget we use depends on whether the clause contains two or three variables (see Figure 2). We restrict our description to the three-variable case, as the two-variable case is similar and simpler. In Figure 2, the vertices marked by unfilled circles will be identified with vertices from a variable gadget. There are again three groups of such vertices, one for each variable occurring in the clause.

Obviously, we will identify the t -vertex of a variable group with the t -vertex of a clause group, and the same for the f -vertices. We call this *matching*. It is not entirely straightforward to do this matching in a manner that preserves planarity. Consider the way in which the groups and the t and f vertices appear on the boundary of the clause gadget. In Figure 2, the pairs appear in order $(t, f), (t, f), (f, t)$ clockwise starting from the top. As illustrated in Figure 3, $(t, f), (f, t), (f, t)$ is also possible. The remaining two alternatives, $(t, f), (t, f), (t, f)$ and $(f, t), (f, t), (f, t)$ are to be avoided. This is accomplished by choosing a variable appearing negatively in the clause and mirroring the corresponding variable gadget around the axis T_1-F (see Figure 1). This does not affect our ability to connect the variable to other clauses.

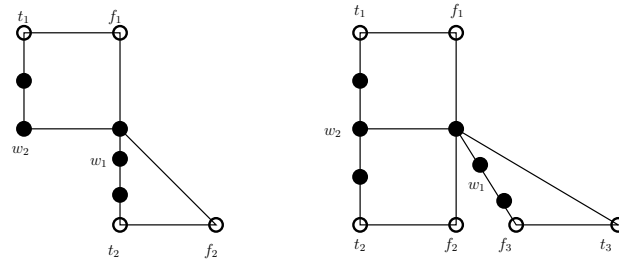


Fig. 3. Alternative planar embeddings of the the clause gadgets.

This completes the construction. Call the resulting graph H_Ψ , which is planar and has bounded degree by construction. We remark that each variable appears once negatively in Ψ , and once or twice positively. So if the variable appears only twice, then (t_1, f_1) or (t_3, f_3) in the corresponding variable gadget will not be identified with a group of vertices in a clause gadget.

In Figure 4 we can see an example of the reduction and the resulting planar graph from the specific instance of 1-NEGATIVE PLANAR 3-SAT $(\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee \neg x_3) \wedge (\neg x_4 \vee x_3)$.

We now make several observations about the graph H_Ψ and the way landmarks need to be positioned on it.

Each f -vertex is contained in a triangle, say with other vertices r, s . Observe that r and s can only be resolved by a landmark on r or s . We call these *forced landmarks*. In fact, in any smallest set of landmarks exactly one of r, s will be a landmark. Then it follows by construction that H_Ψ requires exactly $3n$ forced landmarks, where n is the number of variables of Ψ .

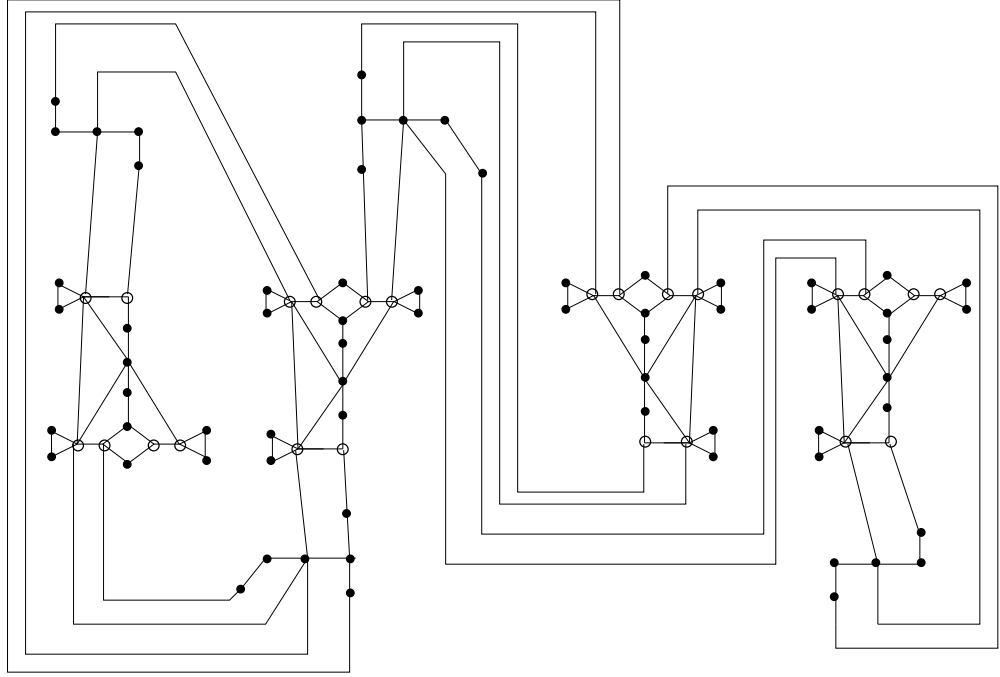


Fig. 4. The planar graph obtained for $(\neg x_1 x_2)(x_1 \neg x_2 x_3)(x_2 x_4 \neg x_3)(\neg x_4 x_3)$

Using the forced landmarks, we can resolve most pairs of vertices, as shown by the following lemma. We say that T_1, T_2, N_1, N_2 and F are *strictly inside* the variable gadget.

Lemma 4 *Let x, y be a pair of vertices, which is not $\{w_1, w_2\}$, $\{T_1, T_2\}$, $\{T_1, N_1\}$ or $\{T_2, N_1\}$ from the same gadget. Then the pair is resolved by forced landmarks.*

Proof. It is straightforward to check the cases where both x and y are in the same clause or variable gadget. There are two remaining cases: either x, y are in different clause gadgets, or x is strictly inside a variable gadget and y outside that gadget.

Consider the first case, that is, x, y are in different clause gadgets. Denote the gadget containing x by g_x , and the gadget containing y by g_y . Let z_x be a forced landmark that is closest to x , and let z_y be a forced landmark that is closest to y . Without loss of generality, $d(x, z_x) \leq d(y, z_y)$. We will show that $d(z_x, x) < d(z_x, y)$. Since x, y are in distinct clause gadgets, the shortest path P from y to z_x crosses at least one variable gadget g , and enters it through a group (t, f) . Let w be the first vertex of $\{t, f\}$ that occurs on P , and let z_f be the forced landmark in the triangle connected to f . If $w = t$, then P contains at least two edges in g , and if $w = f$, P contains at least one edge in g . In either case P also contains at least one edge in g_x . Since $d(z_f, f) = 1$, $d(z_f, t) = 2$, in both cases the inequality $d(w, z_f) < d(w, z_x)$ holds. Hence $d(x, z_x) \leq d(y, z_y) \leq d(y, z_f) \leq d(y, w) + d(w, z_f) < d(y, w) + d(w, z_x) = d(y, z_x)$.

Now consider the second case, and assume that x is strictly inside the variable gadget in Figure 5. If y is in the picture, it can be readily verified that x and y are resolved by the forced landmarks of the variable gadget. We claim that if y is outside of the picture, then $d(z_1, y) + d(z_3, y) \geq 7$, where z_1 and z_3 are the forced landmarks in the triangles attached to f_1 and f_3 respectively. This implies that z_1 or z_3 is at distance at least four from y , whereas the distance of z_1 and z_3 to x is at most three, implying that x and y are resolved.

To prove the claim, note that if shortest paths from y to z_1 and z_3 both contain f_1 (or equivalently f_3), then $d(z_1, y) + d(z_3, y) = d(z_1, f_1) + d(z_3, f_1) + 2d(f_1, y) = 4 + 2d(f_1, y) > 6$. Now consider the case where shortest paths from y to z_1, z_3 include f_1, f_3 respectively. Any path from f_1 to f_3 that is not contained in the figure must cross at least two clause gadgets and one variable gadget, so it has length at least 5. This gives $d(y, f_1) + d(y, f_3) \geq 5$, and $d(z_1, y) + d(z_3, y) \geq 7$.

It remains to analyze how the pairs excluded in Lemma 4 can be resolved. This will rely on the satisfiability of Ψ of course (as described below), but the following auxiliary lemma is crucial.

Lemma 5 *All pairs of vertices that are strictly inside a variable gadget are resolved if and only if there is a landmark strictly inside the variable gadget.*

Proof. It is easy to check that a landmark that is strictly inside a variable gadget together with the forced landmarks resolves all pairs of vertices that are strictly inside the gadget. If no landmark is strictly inside the variable gadget, then a shortest path from any landmark z to T_1 or T_2 contains t_1 or t_3 . But then $d(z, T_1) = d(z, T_2)$.

This lemma and the forced landmarks together imply that H_Ψ has metric dimension at least $4n$. With this fact in mind, we present the proof of our main result.

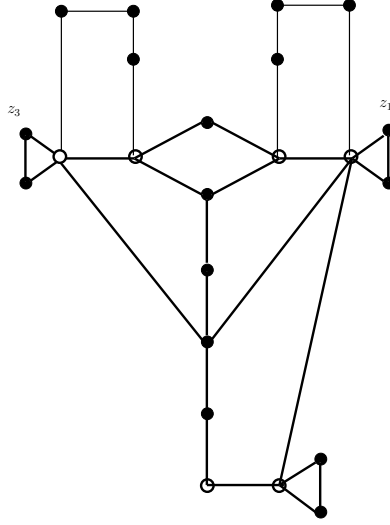


Fig. 5. The subgraph in Lemma 4.

Theorem 6 PLANAR METRIC DIMENSION is NP-complete, even on bounded-degree graphs.

Proof. Let Ψ be an instance of 1-NEGATIVE PLANAR 3-SAT with n variables. Construct the graph H_Ψ in the manner described before. Constructing H_Ψ clearly takes time polynomial in the number of variables and clauses of Ψ . We now claim that H_Ψ has metric dimension $4n$ if and only if Ψ is satisfiable.

Assume that a satisfying truth assignment for Ψ is given. Place $3n$ forced landmarks. If a variable has value true, place a landmark on T_1 in the corresponding gadget; otherwise, place a landmark on F . After applying Lemmata 4 and 5, we only need to check that pairs w_1, w_2 are resolved. But each such pair is resolved by the landmark strictly inside the variable that satisfies the corresponding clause, and we are done.

Now assume that the metric dimension is $4n$. We will construct a satisfying assignment for Ψ . Each variable gadget contains exactly one landmark, which is on T_i , N_i , or F . If the landmark is on T_i , set the variable to true. If the landmark is on F , set it to false. Otherwise the variable can be arbitrarily set to either true or false. It remains to show that because the pairs w_1, w_2 are resolved, the truth assignment is satisfying. Note that a landmark z resolves pair w_1, w_2 if a shortest path from a landmark to either of them enters the clause gadget through some t_i . If a shortest path from landmark z to w_1 or w_2 intersects more than one clause gadget, it leaves the first clause through an f -vertex, after which it enters all subsequent ones through an f -vertex. But then w_1, w_2 in the final clause are not resolved. It follows that a landmark z resolves w_1 and w_2 only if

it is in an adjacent variable gadget and the corresponding variable satisfies the corresponding clause.

Following the claim, the reduction should construct H_Ψ as the graph for the instance of PLANAR METRIC DIMENSION and set k to $4n$.

3 Algorithm for Outerplanar Graphs

In this section, we prove that METRIC DIMENSION can be solved in polynomial time on outerplanar graphs. So let G be an outerplanar graph, given together with an outerplanar embedding. Note that a metric base of a disconnected graph is the union of metric bases of its component (with one exception: isolated vertices, an edgeless graph of n vertices has metric dimension $n - 1$). So we can safely assume that G is a connected graph. We can also assume that it has at least three vertices.

As a technical trick we sometimes treat the midpoint of an inner edge $e = (v_1, v_2) \in E(G)$ as an actual vertex. The distances from this *midpoint vertex* v_e are such that $d(v_e, v_1) = d(v_e, v_2) = \frac{1}{2}$ and $d(v_e, x) = \min(d(v_e, v_1) + d(v_1, x), d(v_e, v_2) + d(v_2, x))$.

3.1 Characterization of Resolving Sets in Outerplanar Graphs

We first give a characterization of the effects of resolving sets in vertices and faces of outerplanar graphs. To this end, we introduce some notation. A *bifurcation point* associated with z, x, y is a vertex v farthest from z such that v is on shortest path from z to both x and y . More formally, v is a bifurcation point if it is on shortest paths $z \rightsquigarrow x$, $z \rightsquigarrow y$, and if any shortest paths $v \rightsquigarrow x$, $v \rightsquigarrow y$ intersect only in v . Notice that in an outerplanar graph the bifurcation point for each triple of vertices is unique. We define the function $g : V(G) \times \mathcal{P}(V(G)) \rightarrow \mathcal{P}(V(G))$ as

$$g(v, L) = \{w \in \mathcal{N}(v) : d(z, w) = d(z, v) + 1 \text{ for all } z \in L\}.$$

In other words, a neighbor w of v is in $g(v, L)$ if for every $z \in L$, v is on some shortest path $z \rightsquigarrow w$ (but not necessarily on every shortest path.)

Any pair $x, y \in g(v, L)$ is left unresolved by landmarks in L . So any resolving set L satisfies the following:

Requirement 1 *Any vertex $v \in V(G)$ must have $|g(v, L)| \leq 1$.*

If G is a tree, then the requirement is sufficient. The following lemma gives an alternative, simpler correctness proof for the algorithm by Khuller et al. [16].

Lemma 7 *If G is a tree with at least three vertices, then a set $L \subseteq V(G)$ is a resolving set if and only if it satisfies Requirement 1.*

Proof. We have already seen that any resolving set L satisfies Requirement 1. Now assume that Requirement 1 is satisfied. We pick any two vertices $x, y \in V(G)$ and show that they are resolved.

Since G has at least 3 vertices, there is at least one vertex $v \in V(G)$ with degree at least 2. Since $|g(v, L)| \leq 1 < |\mathcal{N}(v)|$, L is not empty.

Choose any landmark $z \in L$. If z resolves x, y , we are done. Otherwise, let v be the bifurcation point associated with z, x, y , and let v_1, v_2 be the successors of v on the shortest paths $v \rightsquigarrow x, v \rightsquigarrow y$ (see Fig. 6). Since $d(z, x) = d(z, y)$, we have $d(v, x) = d(v, y)$. By assumption, $g(v, L)$ can not contain both v_1 and v_2 . Without loss of generality $v_1 \notin g(v, L)$. Then there is a landmark z_2 with $d(z_2, v_1) < d(z_2, v)$, and furthermore $d(z_2, x) < d(z_2, y)$. \square

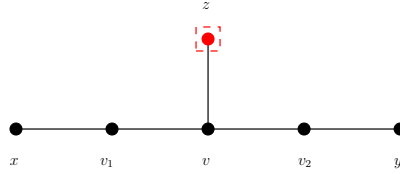


Fig. 6. Proof of Lemma 7.

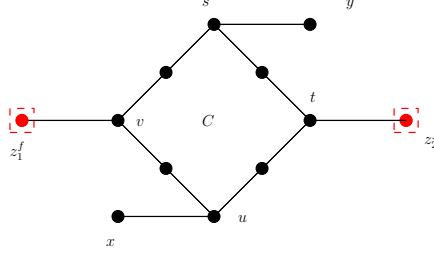
As stated earlier, the major difficulty of the metric dimension problem is that it is non-local. This is why Lemma 7 is useful. Although stopping short of giving an actual local characterization of resolving sets, it does make the effects of a resolving set local enough to devise a polynomial-time algorithm for trees.

Our algorithm relies on a generalization of Lemma 7 to outerplanar graphs. In this case Requirement 1 no longer implies that L is a resolving set. For example, if G is an even cycle and L contains two antipodal vertices, then Requirement 1 is satisfied, but L is not a resolving set. Therefore we need another requirement that provides an additional condition for the effects of a resolving set on the faces of outerplanar graphs. First, we need some auxiliary definitions and lemmas.

Definition 8 *Let $z \in V(G)$ be a landmark, and let C be either a single edge or a cycle—in particular, it may be a face. The representative of z on C is the element of $V(C)$ closest to z , if it is unique. Otherwise outerplanarity implies that there are two closest vertices, which are adjacent. In this case the representative is the midpoint of those two vertices.*

We can make the following observations. Let C, C' be cycles such that $V(C') \subseteq V(C)$ and C' is a face. If two vertices have the same representative on C , then they must have the same representative on C' as well.

Lemma 9 *Let G be a graph, and let $x, y, z, z_2 \in V(G)$. If neither z nor z_2 resolves the pair x, y and there exist intersecting shortest paths $z \rightsquigarrow x, z_2 \rightsquigarrow y$, then a bifurcation point of z, x, y is also a bifurcation point of z_2, x, y .*

**Fig. 7.** Requirement 2.

Proof. Assume that w is a vertex which is both on some shortest path $z \rightsquigarrow x$ and some shortest path $z_2 \rightsquigarrow y$. Let v be a bifurcation point of w, x, y . Then there exist shortest paths $z \rightsquigarrow x$, $z_2 \rightsquigarrow y$ both containing both v and w . We need to show that v is also in the intersection of some pair of shortest paths $z \rightsquigarrow y$, $z_2 \rightsquigarrow x$.

The proof relies on the fact that the equality $d(a, c) = d(a, b) + d(b, c)$ holds if and only if b is on a shortest path from a to c .

By the triangle inequality and since neither z nor z_2 resolve x, y ,

$$\begin{aligned} d(z, y) + d(z_2, x) &\leq (d(z, v) + d(v, y)) + (d(z_2, v) + d(v, x)) \\ &= (d(z, v) + d(v, x)) + (d(z_2, v) + d(v, y)) \\ &= d(z, x) + d(z_2, y) \\ &= d(z, y) + d(z_2, x). \end{aligned}$$

Therefore equality holds in the triangle inequality, and the claim follows. \square

We are now ready to present the other necessary requirement.

Requirement 2 Let v' be a face of an outerplanar graph G and let $L \subseteq V(G)$ have exactly two representatives \hat{z}_1, \hat{z}_2 on v' . Let z_1^f and z_1^l be the landmarks with representative \hat{z}_1 which occur first and last on the walk along the outer face starting at \hat{z}_2 , and define z_2^f, z_2^l analogously. Assume that neither x_1^f nor x_2^f resolves x, y , and that shortest paths $z_1^f \rightsquigarrow x$, $z_2^f \rightsquigarrow y$ do not intersect. Let v be the bifurcation point of z_1^f, x, y , let u be the bifurcation point of x, z_1^f, z_2^f , etc., as in Fig. 7. By assumption, $v \neq t$. Therefore the shortest paths $s \rightsquigarrow t$, $t \rightsquigarrow u$, $u \rightsquigarrow v$, $v \rightsquigarrow s$ form a cycle C (see again Fig. 7). If v' is inside the cycle C , then one of z_1^l, z_2^l must resolve the pair x, y .

Note that the representatives of z_1^f and z_2^f on C are v and t , respectively.

The assumption that G is outerplanar is essential for Requirement 2 and Lemma 10 below. In particular, the definition of $z_1^f, z_2^l, z_2^f, z_1^l$, as well as the use of representatives in the proof of Lemma 10, relies on outerplanarity.

Lemma 10 If G is an outerplanar graph, then any resolving set $L \subseteq V(G)$ satisfies Requirement 2.

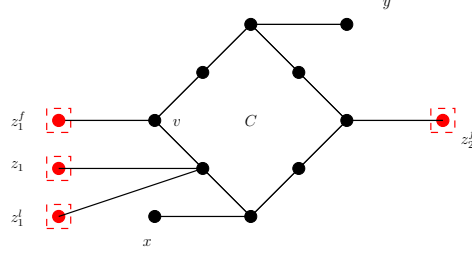


Fig. 8. Neither of z_1 , z_1^l has representative v on C .

Proof. Assume that L is a resolving set, and that v', x, y are as in Requirement 2. Without loss of generality, there is a $z_1 \in L$ with representative \hat{z}_1 on v' such that $d(z_1, x) < d(z_1, y)$. Now define z_1^f, z_2^f, z_1^l as in Requirement 2. Assume that neither z_1^f nor z_2^f resolves x and y . Then we can define v, t, C as in Requirement 2. Observe that the vertices v and t are the only vertices of C that do not resolve x and y . Therefore, while landmark z_1^f has representative v on C , z_1 does not. Because of the location of z_1^f, z_1, z_1^l on the outer cycle, the representative of z_1^l on C is not v either (see Fig. 8). The representative of z_1^l on C cannot equal t , as z_1^l and z_2^f have distinct representatives on v' . Therefore the representative of z_1^l resolves x and y , and so does z_1^l itself. \square

Now we are ready to generalize Lemma 7 to outerplanar graphs. This is a crucial result, since it characterizes resolving sets in a manner that allows dynamic programming.

Theorem 11 *If G is an outerplanar graph, then a set $L \subseteq V(G)$ is a resolving set if and only if it satisfies Requirements 1 and 2.*

Proof. We have already seen that any resolving set satisfies the requirements. So assume that $L \subseteq V(G)$ satisfies the Requirements, and choose any $x, y \in V(G)$. We show that there exists a $z \in L$ that resolves the pair x, y .

As in Lemma 7, L is non-empty. Choose $z \in L$ arbitrarily. If z resolves x and y , we are done; so assume that it does not. As in Lemma 7, let v be the bifurcation point of z, x, y , and let v_1, v_2 be successors of v on some shortest paths $v \rightsquigarrow x, v \rightsquigarrow y$ respectively. By Requirement 1, there is a $z_2 \in L$ such that, without loss of generality, $d(z_2, v_1) \leq d(z_2, v)$. If z_2 resolves x and y , we are done. So assume otherwise.

If there exist intersecting shortest paths $z \rightsquigarrow x, z_2 \rightsquigarrow y$, then by Lemma 9, v is on a shortest path $z_2 \rightsquigarrow x$. Then v_1 is also on such a path, and $d(z_2, v_1) = d(z_2, v) + 1$, a contradiction. Therefore no such pair of intersecting shortest paths exists. Define v, t, C as in Requirement 2, and let v' be a face inside C . If there exists a $z_3 \in L$ whose representative on v' is distinct from the representatives of z_1 and z_2 , then its representative on C is neither v nor t . Hence z_3 resolves x, y . If such a landmark z_3 does not exist, then Requirement 2 implies the claim. \square

3.2 Generalized dual tree

We now introduce first of the the data structures that we need in our algorithms. The *weak dual* of an outerplanar graph G is a graph which has the faces of G , except the unbounded outer face, as vertices. Two faces are adjacent if they share an edge. The weak dual of a biconnected outerplanar graph is a tree.

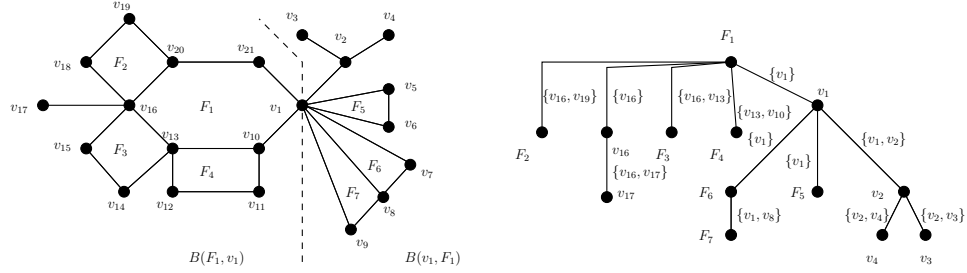


Fig. 9. An outerplanar graph G and its generalized dual tree T . The blue dashed line indicates how G is divided to $B(F_1, v_1)$ and $B(v_1, F_1)$. The sets on the edges of T indicate the values of $s(e')$.

The *generalized dual tree* $T = (V', E')$ of an outerplanar graph G is defined as follows. T contains the weak dual of G , and also contains the subgraph of G induced by all cut vertices and vertices of degree one. For any cut vertex contained in a biconnected component, there is an edge from the vertex to an arbitrary incident face of the component. Observe that the resulting graph is a tree (see Fig. 9 for an example). According to this definition, a cut vertex is a vertex of both G and T . Let any vertex of T be the root, denoted by v'_r .

We now associate a subset of $V(G)$ with any vertex or edge of T . If $v' \in V(T)$ is a face, the set $s(v')$ consists of the vertices on the face. If v' is a cut vertex or a leaf, $s(v')$ consists of that vertex. If both endpoints of $e' \in E(T)$ are vertices of G , then $s(e')$ consists of those vertices. Otherwise, let either endpoint of e' is a face. Let $e' = (v'_1, v'_2)$, and define $s(e') = s(v'_1) \cap s(v'_2)$.

Removing an edge (v', w') divides T into two components, $T_{v'}$ and $T_{w'}$, where $T_{v'}$ is the one containing v' . Define $B(v', w')$ as the subgraph of G corresponding to $T_{v'}$. Formally, it is the subgraph of G induced by $\bigcup_{u' \in V(T_{v'})} s(u')$. Thus G is divided into two subgraphs $B(v', w')$, $B(w', v')$. If v', w' are adjacent faces, the subgraphs share two vertices. If v' is a face and w' a cut vertex (or the other way around), then the subgraphs share one vertex. Otherwise they do not intersect. Define $B^-(v', w')$ as the subgraph of G induced by $V(G) \setminus V(B(w', v'))$. Then we can divide G into two nonintersecting subgraphs, $B^-(v', w')$ and $B(w', v')$.

The following lemma is immediate from the definitions.

Lemma 12 *Given neighbors $v', w' \in V(T)$, $B(v', w')$ and $B(w', v')$ are connected subgraphs of G , and any path from $B(v', w')$ to $B(w', v')$ intersects $s((v', w'))$.*

Given a landmark z , let $S(z, W) = \{\{x, y\} \in W \times W : d(z, y) \neq d(z, x)\}$ be the set of pairs in $W \subseteq V(G)$ resolved by z .

Lemma 13 *Let $e' = (v', w') \in E(T)$. Assume $z \in V(B(v', w'))$ and denote $W = V(B(w', v'))$. Then $S(z, W) = S(\hat{z}, W)$, where \hat{z} is the representative of z on e' .*

Proof. Let $z \in V(B(v', w'))$ and $x \in W$. If $\hat{z} = v_1$ or v_2 , then $d(z, x) = d(z, \hat{z}) + d(\hat{z}, x)$. If $\hat{z} = v_e$, then $d(z, x) = d(z, \hat{z}) + d(\hat{z}, x) - 1$. So for any $x, y \in W$, we have $d(z, x) - d(z, y) = d(\hat{z}, x) - d(\hat{z}, y)$. \square

Lemma 14 *Let $e' = (v', w') \in E(T)$ be an edge. If e' corresponds to an edge $e = (v_1, v_2) \in E(G)$ with midpoint v_e , then $S(v_e, W) \subseteq S(v_1, W) \cup S(v_2, W)$.*

Proof. Assume that $\{x, y\} \in S(v_e, W) \setminus S(v_1, W)$. So, we have

$$d(v_1, x) = d(v_1, y) \text{ and } d(v_e, x) \neq d(v_e, y)$$

Without loss of generality, we assume that $d(v_e, x) < d(v_e, y)$. Then

$$\frac{1}{2} + \min(d(v_1, x), d(v_2, x)) < \frac{1}{2} + \min(d(v_1, y), d(v_2, y)).$$

Putting these equations together implies that $d(v_2, x) < d(v_2, y)$, that is, $\{x, y\} \in S(v_2, W)$. \square

3.3 Boundary conditions

Let v' be a dual vertex and p' its parent. We use *boundary conditions* to describe the relation of landmarks in $B(v', p')$ to $B(p', v')$, and vice versa. Boundary conditions can be seen as an abstract data structure which depends on v', p', L and satisfies the following:

1. It consists of two components, one of which depends on landmarks in $B(p', v')$, and the other on landmarks in $B^-(v', p')$. The components are called *upper* and *lower* boundary conditions, respectively.
2. It determines which pairs in $B(p', v')$ are resolved by landmarks in $B^-(v', p')$, and vice versa.
3. If $B(v', p')$ contains landmarks with a representative v on v' , then the boundary condition specifies which such landmarks occur first and last on a walk along the outer boundary of $B(v', p')$ starting at v .
4. For any $v \in s((p', v'))$, it specifies whether the set $g(v, L) \cap V(B^-(v', p'))$ is empty or not.
5. The number of possible boundary conditions is polynomial.

The first and the second properties are necessary to be able to run a dynamic programming algorithm along T . The third and fourth properties are needed when verifying Requirements 2 and 1, respectively. The last property is needed to ensure that the algorithm runs in polynomial time.

Let us consider the implementation of this data structure. Let $e' = (p', v') \in E(T)$, $s(e') = \{v_1, v_2\}$. The boundary condition of e' is a tuple

$$\mathbf{t} = (t_u, t_l, t_z, t_{v_1}, t_{v_2}) \in \mathcal{P}(\{v_1, v_2, v_e\})^2 \times \mathcal{P}(V(G)) \times \{0, 1, 2\}^2.$$

The set t_u consists of the representatives of $L \cap V(B(p', v'))$ on e' , and t_l consists of the representatives of $L \cap V(B^-(v', p'))$ on e' .

The set t_z consists of all the first and last landmarks with given representative as property 3 in above requires. Note that t_l is technically redundant, since t_z determines its value. As an optimization, t_z could be omitted in some cases.

The value of t_{v_1} depends on the properties of the set

$$V_1 = g(v_1, L \cap V(B^-(v', p'))) \cap V(B^-(v', p')).$$

If V_1 contains a vertex x which is not a neighbor of v_2 , then $t_{v_1} = 0$. If $V_1 = \{x\}$ with $x \in \mathcal{N}(v_1) \cap \mathcal{N}(v_2)$, then $t_{v_1} = 1$. Otherwise $V_1 = \emptyset$ and $t_{v_1} = 2$. We define t_{v_2} similarly, with the roles of v_1 and v_2 interchanged.

Lemma 15 *The implementation of boundary conditions described above has properties 1–5 given above.*

Proof. A boundary condition $(t_u, t_l, t_z, t_{v_1}, t_{v_2})$ can be divided into an upper boundary condition t_u and a lower boundary condition $(t_l, t_z, t_{v_1}, t_{v_2})$, so property 1 holds. Property 2 follows from Lemma 13. Property 3 follows from the definition of t_z .

To verify that property 4 is satisfied, consider $g(v_1, L) \cap V(B^-(v', p'))$. This set is empty if either $t_{v_1} = 1$ and $v_2 \in t_u$, or if $t_{v_1} = 2$. Otherwise, the set is nonempty. We can interchange v_1 and v_2 , and the same holds. So property 4 is satisfied.

It remains to verify property 5 is satisfied. There are (at most) $O(|V|)$ edges e' , and for each such edge there are $2^3 - 1$ possible values of t_u , $2^3 - 1$ possible values of t_l , 3 possible values of t_{v_1} , 3 possible values of t_{v_2} , and $O(|V|^6)$ for t_z . Therefore property 5 holds. \square

3.4 Configurations

While a boundary condition describes how the landmarks are placed in relation to a dual edge e' , a *configuration* describes their relation to a dual vertex v' . There are three quite different cases, depending on whether v' is a cut vertex of G , a face with two representatives, or a face with at least three representatives. The last-mentioned case is quite complicated.

A configuration L associated with v' is designed to satisfy following:

1. For any $w' \in \mathcal{N}(v')$, it determines which pairs of vertices in $B(w', v')$ are resolved by landmarks in $B(v', w')$.
2. It contains enough information to verify that v' satisfies Requirement 2.
3. The total number of configurations is polynomial.

Essentially, a configuration determines which vertices of $s(v')$ are landmarks, and which boundary conditions are allowed for edges (v', w') .

Implementation of this data structure starts by extending the concept of representative to take into account the branch of the dual that contains the landmark yielding the representative. Formally, the *extended representative* of $z \in L$ on v' is a pair (\hat{z}, w') . Here \hat{z} is the representative of z on v' . Moreover, if $z \in s(v')$, then $w' = v'$; otherwise, w' is the neighbor of v' such that $z \in V(B^-(w', v'))$.

Given a face v' and vertex $\hat{z} \in \tilde{s}(v')$, there are at most four values of w' such that (\hat{z}, w') is a valid extended representative, namely v' , at most two neighboring faces, and at most one neighboring cut vertex. For example, in Fig. 9, if $\hat{z} = v_{16}$, w' can be F_1 , F_2 , F_3 , or v_{16} .

We can now define the configurations for each of the three cases.

Definition 16 *Let L be a set of landmarks and v' a dual vertex which corresponds to a cut vertex $v \in V(G)$. Let R consist of all extended representatives of L on v' . The configuration associated with v', L is the pair (v', R) . This is said to be a configuration of type I.*

In this case, any extended representative has pattern (v, w') with $w' \in \mathcal{N}(v') \cup \{v'\}$. Hence the total number of configurations of type I would be $2^{(\deg(v')+1)}$. However, from here on we only consider the ones that fulfill Requirement 1. Then there is at most one $w' \in \mathcal{N}(v')$ such that $(v, w') \notin R$. Therefore the total number of configurations of type I is $O(|E(T)|) = O(|V|)$.

Definition 17 *Let L be a set of landmarks and v' a face such that L has two representatives on v' . Let $R = \{z_1^f, z_1^l, z_2^f, z_2^l\}$, where the landmarks are defined as in Requirement 2. Let the configuration associated with v' and L be the pair (v', R) . This is said to be a configuration of type II.*

In this case the number of configurations clearly is $O(|V|^4)$.

The remaining case, a face with more than two representatives, is the most intricate because of combinatorial explosion: on a face of n vertices, there are up to 3^n possible sets of representatives.

Definition 18 *Let L be a set of landmarks and v' a face such that L has more than two representatives on v' . The configuration associated with v', L is a pair (v', R) , where R is computed by the (nondeterministic) Algorithm 1. This is said to be a configuration of type III.*

As an example of a configuration of type III, consider Fig. 10.

Note that Algorithm 1 is a part of a mathematical definition. It is not actually used in any computations. Algorithm 1 is given a set of landmarks and then produces a configuration. In our computations to find a metric base (Algorithm 4), we proceed in the opposite direction: we iterate over pairs (v', R) , and for each of them, we produce a set of landmarks, if possible. The following lemma shows that we can restrict to sets R such that $|R| \leq 20$, and furthermore

Algorithm 1 Construct-Conf**Input:** Dual vertex v' , set L

-
- 1: Let \hat{L} be the set of representatives of L on v' .
 - 2: Choose $\hat{z}_1, \hat{z}_2 \in \hat{L}$ at maximal mutual distance.
 - 3: Choose $\hat{z}_3 \in \hat{L} \setminus \{\hat{z}_1, \hat{z}_2\}$ which is, if possible, not on a shortest path $\hat{z}_1 \rightsquigarrow \hat{z}_2$.
 - 4: $R_1 \leftarrow \{\hat{z}_1, \hat{z}_2, \hat{z}_3\}$
 - 5: **if** the shortest path $\hat{z}_1 \rightsquigarrow \hat{z}_2$ is not unique and both paths contain a vertex of $\hat{L} \setminus \{\hat{z}_1, \hat{z}_2\}$ **then**
 - 6: Choose $\hat{z}_4 \in \hat{L} \setminus \{\hat{z}_1, \hat{z}_2\}$ which is on a different shortest path than \hat{z}_3
 - 7: $R_1 \leftarrow R_1 \cup \{\hat{z}_4\}$
 - 8: **end if**
 - 9: **for all** $i, j \in \{1, 2, 3\}$ **do**
 - 10: **if** $d(\hat{z}_i, \hat{z}_j) = (s(v') - 1)/2$ and there is $\hat{z}_n \in \hat{L} \setminus R_1$ which is on the shortest path $\hat{z}_i \rightsquigarrow \hat{z}_j$ **then**
 - 11: $R_1 \leftarrow R_1 \cup \{\hat{z}_n\}$
 - 12: **end if**
 - 13: **end for**
 - 14: Let R consist of all extended representative corresponding to elements of R_1
 - 15: **return** R
-

the set $Q = \{\hat{z} : (\hat{z}, w') \in R \text{ for some } w'\}$ must satisfy $3 \leq |Q| \leq 5$. Because $|R| \leq 20$, the total number of configurations of type III that we need to iterate over in our computations thus is $O(|V|^{20})$.

Lemma 19 *Algorithm 1 produces a set R with $|R| \leq 20$, such that R determines the upper boundary condition of (w', v') for any child w' of v' . If v' has parent p' , R also determines the component t_l of the lower boundary condition of (p', v') .*

Proof. We first show that $|R| \leq 20$. Initially, R_1 will consist of three elements. Either line 6 adds one more element, or the loop on lines 9–11 adds at most two elements. So $|R_1| \leq 5$, and $|R| \leq 4|R_1| = 20$.

We now verify the effect of R on the boundary conditions described in the lemma statement. Observe that, by Lemma 13, any two landmarks with the same extended representative have the same effect on the boundary conditions in Definition 18. Therefore it is indeed justified to talk about the upper/lower boundary condition determined by R .

We first prove the following auxiliary claim.

Claim 1: Any $\hat{z} \in \hat{L}$ is on a shortest path $\hat{z}_i \rightsquigarrow \hat{z}_j$ for some $i, j \in \{1, 2, 3\}$.

Proof: If every representative is in shortest a path $\hat{z}_1 \rightsquigarrow \hat{z}_2$, the claim holds. Otherwise, \hat{z}_3 is selected so that it is not in the shortest path $\hat{z}_1 \rightsquigarrow \hat{z}_2$. Then shortest path $\hat{z}_1 \rightsquigarrow \hat{z}_2$ and $\hat{z}_1 \rightsquigarrow \hat{z}_3$ intersect only in \hat{z}_1 , because otherwise $d(\hat{z}_1, \hat{z}_3) > d(\hat{z}_1, \hat{z}_2)$, contradicting the choice of \hat{z}_1 and \hat{z}_2 . Also shortest paths $\hat{z}_2 \rightsquigarrow \hat{z}_1$, $\hat{z}_2 \rightsquigarrow \hat{z}_3$ intersect only in \hat{z}_2 . Hence the shortest paths cover all vertices in $s(v')$. This proves the claim. \square

We now prove that R determines the upper boundary condition of (w', v') for any child w' of v' . So let w' be a child of v' , let $e' = (v', w')$, and let

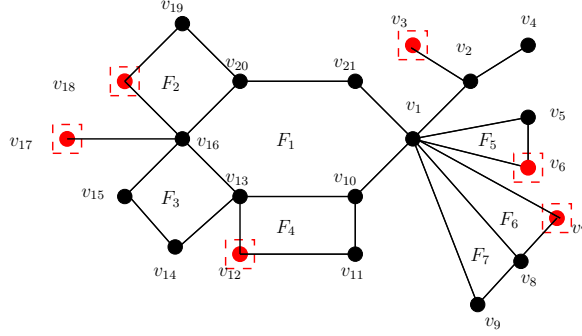


Fig. 10. Resolving set $L = \{v_3, v_6, v_7, v_{15}, v_{17}, v_{18}\}$ has a configuration (F_1, R) with $R = \{(v_{16}, v_{16}), (v_{16}, F_2), (v_{13}, F_4), (v_1, v_1)\}$. Values of the function h are $h(v_1, F_1, L) = v_1$ (because $g(v_1, L) = \{v_9\} \subset V(B^-(v_1, F_1))$), $h(v_{10}, F_1, L) = \emptyset$, $h(v_{13}, F_1, L) = v_{16}$, $h(v_{16}, F_1, L) = F_3$, $h(v_{20}, F_1, L) = h(v_{21}, F_1, L) = \emptyset$.

$s(e') = \{v_1, v_2\}$. Let t_u be the upper boundary condition of L on e' , and let r_u be the upper boundary condition of R on e' . We need to show that $r_u = t_u$.

The inclusion $v \in r_u$ holds if and only if there is a $(\hat{z}, u') \in R$ such that $u' \neq w'$ and \hat{z} has representative v' on e' . Similarly, $v \in t_u$ if there is $z \in L$ with such an extended representative.

Since R corresponds to a subset of L , $r_u \subseteq t_u$. To show inclusion in the other direction, consider $v \in t_u$. By definition there is $z \in L$ with $z \notin V(B(w', v'))$ that has representative v on e' . Denote the representative of z on v' by \hat{z} . By Claim 1, \hat{z} is on a shortest path $\hat{z}_i \rightsquigarrow \hat{z}_j$, for some $i, j \in \{1, 2, 3\}$. We distinguish several cases.

Case 1: $\hat{z} = v$.

Since $z \notin V(B(w', v'))$, $\hat{z} \neq v_e$, and thus $v \neq v_e$. If one of \hat{z}_i, \hat{z}_j has representative v on e' (say \hat{z}_i), then it must have an extension (\hat{z}_i, u') with $u' \neq w'$ (as in Figure 11.) Therefore $v \in r_u$ and we are done. So suppose that neither has representative v on e' . Then the representative of both \hat{z}_i and \hat{z}_j on e' is v_e (the midpoint of e). By construction, this implies that $\{i, j\} = \{1, 2\}$. In particular, one of \hat{z}_i, \hat{z}_j equals v_e and the other is antipodal to it (see Fig. 13). Then either \hat{z}_3 or \hat{z}_4 has representative v on e' , and we are done.

Case 2: $\hat{z} \neq v$.

We distinguish two subcases.

Case 2a: $v = v_e$.

In this case, either one of \hat{z}_i, \hat{z}_j equals \hat{z} , or not. If one of \hat{z}_i, \hat{z}_j equals \hat{z} , then $v_e \in r_u$, and we are done. Otherwise, $r_u = \{v_1, v_2\}$ and $t_u = \{v_1, v_2, v_e\}$ (see Fig. 12), and the sets are equivalent in the equivalence relation used for boundary conditions.

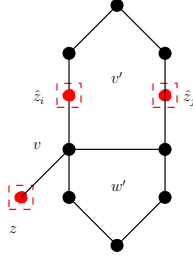


Fig. 11. Illustration for Case 1 in the proof of Lemma 19.

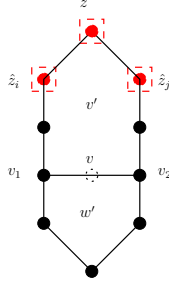


Fig. 12. Illustration for Case 2a in the proof of Lemma 19. Here v is the midpoint.

Case 2b: $v \neq v_e$.

In this case, \hat{z}_i and \hat{z}_j both have representative v_e or v on e' . We again distinguish several subcases.

Case 2b-i: \hat{z}_i and \hat{z}_j both have representative v_e on e' .

In this case, the shortest path $\hat{z}_i \rightsquigarrow \hat{z}_j$ is not unique. Hence either \hat{z}_3 or \hat{z}_4 has representative v on e' (see Fig. 13), and we are done.

Case 2b-ii: Both \hat{z}_i and \hat{z}_j have representative v on e' .

In this case, at least one of them (say \hat{z}_j) has an extension (\hat{z}_j, u') with $u' \neq w'$ (see Figure 14). Hence $v \in r_u$.

Case 2b-iii: \hat{z}_i has representative v on e' , \hat{z}_j has representative v_e on e' .

In this case, either $\hat{z}_i = v$ or not. If $\hat{z}_i = v$, then line 11 of Construct-Conf adds an element to R_1 —the set $\hat{L} \setminus R_1$ is not empty because it contains \hat{z} —and thereby ensures that $v \in r_u$. Otherwise, \hat{z}_i has an extension (\hat{z}_i, u') with $u' \neq w'$ (see Figure 15). Hence $v \in r_u$.

This completes the case analysis. It follows that $r_u = t_u$.

We note that exactly the same arguments hold when w' is not a child of v' but its parent p' . \square

Even if both boundary conditions for edges (v', w'_1) , (v', w'_2) are allowed by a specific configuration, they may contradict each other. This happens when there

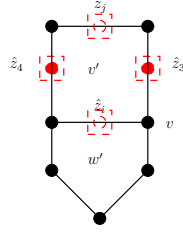


Fig. 13. Illustration for Case 1 and Case 2b-i in the proof of Lemma 19. Both \hat{z}_j and \hat{z}_j have representative v_e on e' .

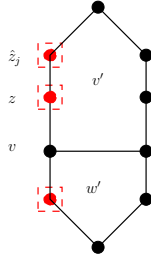


Fig. 14. Illustration for Case 2b-ii in the proof of Lemma 19. Now $\hat{z}_i = v$.

is $v \in s((v', w'_1)) \cap s((v', w'_2))$, and the boundary conditions disagree about the value of $g(v, L)$. Hence, the algorithm will only process boundary conditions that agree with each other and with the configuration.

As a tool for avoiding such disagreements, we define a coarser variant of the function $g(\cdot, \cdot)$. The function $h : V(G) \times V(T) \times \mathcal{P}(V(T)) \rightarrow V(T) \cup \{\emptyset\}$ describes which part of the generalized dual contains $g(v, L)$. Let $v \in s(v')$. Then

$$h(v, v', L) = \begin{cases} v' & \text{if } g(v, L) \cap s(v') \neq \emptyset, \\ w' & \text{if } w' \text{ is neighbor of } v' \text{ and } g(v, L) \cap V(B^-(w', v')) \neq \emptyset, \\ \emptyset & \text{if } g(v, L) = \emptyset. \end{cases}$$

Notice that as long as Requirement 1 holds, the function is well defined.

Lemma 20 *The configuration associated with v' and L determines for any $v \in s(v')$ whether the equation $h(v, v', L) = v'$ holds.*

Proof. Recall that $h(v, v', L) = v'$ if $g(v, L) \cap s(v') \neq \emptyset$. If the configuration (v', R) is of type I, i.e., v' corresponds to a single vertex of G , then $h(v, v', L) = v'$ is not possible.

Otherwise, v' is a face. By Lemma 13, $g(v, L) \cap s(v') = g(v, \hat{L}) \cap s(v')$, where \hat{L} is the set of representatives of L on v' . If (v, R) is of type II, then it determines \hat{L} , and we are done. Hence the configuration is of type III. Let $Q = \{z : (z, w') \in R \text{ for some } w'\}$. Then $Q \subseteq \hat{L}$. We claim that $g(v, L) \cap s(v') = g(v, Q) \cap s(v')$.

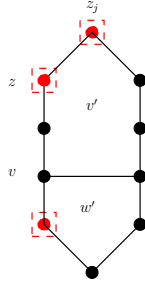


Fig. 15. Illustration for Case 2b-iii in the proof of Lemma 19 in case $\hat{z}_i = v$.

Let $w, v \in s(v')$ be neighbors. If $w \notin g(v, L)$, then there is $z \in L$ with $d(z, w) \leq d(z, v)$. As shown in the proof of Lemma 19, the representative of z on v' is on a shortest path $\hat{z}_1 \rightsquigarrow \hat{z}_2$ for some $\hat{z}_1, \hat{z}_2 \in Q$. Both of them also satisfy $d(\hat{z}_i, w) \leq d(\hat{z}_i, v)$. Hence $w \notin g(v, Q)$.

If $w \notin g(v, Q)$, there is $\hat{z} \in Q$ with $d(\hat{z}, w) \leq d(\hat{z}, v)$. Replace \hat{z} with any vertex $z \in L$ that has representative \hat{z} on v' , and the inequality still holds. Hence $w \notin g(v, L)$. \square

3.5 Algorithm

The algorithm works in a dynamic programming manner, by finding optimal resolving sets of subgraphs $B(v', p')$ with given boundary conditions. Formally, assume that we are given a vertex $v' \in V(T)$ and boundary condition \mathbf{r} on the edge $e' = (v', p')$, where p' is the parent of v' . Let $m(v', \mathbf{t}) \subseteq V(B^-(v', p'))$ be a set $L \cap V(B^-(v', p'))$, where L is a minimal resolving set with boundary condition \mathbf{t} , if such a set exists. Otherwise $m(v', \mathbf{t}) = \text{NIL}$. For notational convenience, we let $|\text{NIL}| = \infty$ and $\text{NIL} \cup A = \text{NIL}$ for any A .

The values of $m(v', \mathbf{t})$ are computed in a recursive manner: the computation of $m(v', \mathbf{t})$ uses the values of $m(w', \mathbf{r})$, where w' is a child of v' . The basic idea is to iterate over all configurations on v' . For every configuration, we then find an optimal function h and an optimal set of landmarks.

First, we introduce several subroutines. Algorithm 2 (Intermediate-sol) is given a configuration (v', R) , a boundary condition \mathbf{t} , and a function h , and it returns an optimal set of landmarks.

Lemma 21 *Algorithm 2 runs in polynomial time and correctly computes an optimal set of landmarks given a configuration (v', R) , a boundary condition \mathbf{t} , and a function h .*

Proof. We first consider the complexity of Algorithm 2. The consistency checks run in linear time. The for-loop runs over $O(|V|)$ neighbors, and for each neighbor, there are $O(|V|^4)$ potential boundary conditions. Hence it runs in polynomial time.

Next we argue that Algorithm 2 is correct. It returns NIL only if a resolving set of $B(v', p')$ with the given parameters does not exist. The algorithm explicitly verifies that Requirement 2 is satisfied. Requirement 1 is verified implicitly by constructing a solution which agrees with h . By assumption the Requirements are satisfied in subgraphs $B(w', v')$. Therefore Lemma 11 states that a non-NIL return value of Algorithm 2 is a resolving set of $B(v', p')$. Assuming that the values of m for all children w' of v' have been computed correctly, the returned set will be an optimal set under the conditions given in the lemma statement. \square

Given a configuration (v', R) and a boundary condition \mathbf{t} , the next subroutine finds an optimal function h . By Lemma 20, the configuration determines whether $h(v, v', L) = v'$ holds or not. Also \mathbf{t} restricts some values of h . Otherwise, the value of $h(v, v', L)$ may be \emptyset or w' , where w' is a suitable neighbor of v' , and the task is to determine which one of these is optimal. It turns out that the optimum can be found by a greedy algorithm (Algorithm 3).

Lemma 22 *Algorithm 3 runs in polynomial time and returns a smallest resolving set of $B(v', p')$ that agrees with the parameters.*

Proof. The for-loops run over $O(|V(T)|)$ cases. The subroutine Intermediate-sol runs in polynomial time. Since $|s((v', w'))| \leq 2$, there is a constant number of possibilities for k in each iteration. Therefore Opt runs in polynomial time.

We need to prove that the function h constructed by Opt is optimal. Let h^o be an optimal function, and define h^i as follows. If $v \notin Q$ after i iterations of the loop in line 10, then $h^i[v] = h[v]$. Otherwise, $h^i[v] = h^o[v]$. Therefore $h^0 = h^o$ and $h^m = h$, where $m = |\mathcal{N}(v')| - 1$.

First note that changing the value of $h[v]$ from some other value to \emptyset increases the value of $\text{Intermediate-sol}(v', R, \mathbf{r}, h)$ by at most one. To see this, let L be the original return value with $g(v, L) = \{x\}$. Then $L_n = L \cup \{x\}$ is an resolving set with $g(v, L_n) = \emptyset$.

To conclude the proof, we show that

$$|\text{Intermediate-sol}(v', R, \mathbf{r}, h^i)| \geq |\text{Intermediate-sol}(v', R, \mathbf{r}, h^{i+1})|.$$

Algorithm 2 Intermediate-sol

Input: Configuration (v', R) , boundary condition \mathbf{t} , function h
if the parameters are inconsistent or (v', R) violates Requirement 2 **then**
 return NIL
end if
Initialize W to the set of landmarks on $s(v')$ as described by (v', R)
for all w' that are children of v' **do**
 $\mathbf{r} \leftarrow \arg \min_{\mathbf{r}} m[w', \mathbf{r}]$ such that \mathbf{r} agrees with C, h, \mathbf{t}
 $W \leftarrow W \cup m[w', \mathbf{r}]$
end for
return $W \cap V(B^-(v', p'))$

Algorithm 3 Opt

Input: Configuration (v', R) , boundary condition \mathbf{t}

$Q \leftarrow \emptyset$ $\{Q$ is the set of vertices for which $h[v]$ is already fixed}

for all $v \in s(v')$ **do**

if R or \mathbf{r} determine $h(v, v', L)$ **then**

$h[v] \leftarrow$ the appropriate value

$Q \leftarrow Q \cup \{v\}$

else

$h[v] \leftarrow \emptyset$

end if

end for

for all w' that are children of v' , in clockwise order, starting on the successor of p' **do**

$P \leftarrow s((v', w')) \setminus Q$

 Find k that minimizes $|\text{Intermediate-sol}(v', R, \mathbf{r}, k)|$ such that

$h[v]$ and $k[v]$ differ only for $v \in P$. If possible, choose a solution with $k[v_i] = \emptyset$ for the last (in clockwise order) element $v_i \in P$

$h \leftarrow k$

$Q \leftarrow Q \cup \{v : h[v] \neq \emptyset\}$

end for

return $\text{Intermediate-sol}(v', R, \mathbf{r}, h)$

Let v_l be as in the algorithm and consider the $(i + 1)$ -th iteration of the second for-loop in Opt. Observe that h^i and h^{i+1} differ only in $s((w', v')) \setminus Q$. If they differ in $v_2 \neq v_l$, then $h^i[v_2] = \emptyset$ and changing the value to $h^{i+1}[v_2]$ does not increase the target value. Then, if $h^i(v_l) \neq h^{i+1}(v_l)$, we first replace $h^i(v_l)$ with \emptyset and then with $h^{i+1}(v_l)$. By the argument above, the first change may increase the size by at most one. The second change must decrease the size, because otherwise the optimization made in line 12 would not change $h[v_l]$. Therefore the changes do not increase the size of the corresponding resolving set. \square

Finally, we are ready to present the main algorithm (Algorithm 4) and its correctness proof.

Theorem 23 *Algorithm 4 correctly computes the values of $m[v', \mathbf{r}]$, and returns a metric base in polynomial time.*

Proof. The algorithm runs in polynomial time, because there is a polynomial number of dual vertices, boundary conditions, and configurations.

We prove by induction that $m[v', \mathbf{r}]$ is computed correctly. Assume that the values of $m[w', \mathbf{t}]$ have been computed correctly for any child w' of v' . Then, because Opt (Algorithm 3) works correctly, the value of $m[v', \mathbf{r}]$ will be computed correctly. Similarly, the return value will be computed correctly, since Opt works correctly and uses correct values of $m[w', \mathbf{r}]$. \square

Algorithm 4 Metric-D

Input: Graph G , its generalized dual tree T

```

for all  $v' \in V(T) \setminus \{v_r'\}$ , boundary condition  $\mathbf{r}$  do {recall that  $v_r'$  is the root}
     $m[v', \mathbf{r}] \leftarrow \text{NIL}$ 
end for
for all  $v' \in V(T) \setminus \{v_r'\}$ , in postorder do
     $p' \leftarrow$  the parent of  $v'$ 
    for all configuration  $(v', R)$  do
        for all boundary condition condition  $\mathbf{r}$  on  $(v', p')$  do
            if  $|\text{Opt}(v', R, \mathbf{r})| \leq |m[v', \mathbf{r}]|$  then
                 $m[v', \mathbf{r}] \leftarrow \text{Opt}(v', R, \mathbf{r})$ 
            end if
        end for
    end for
end for
 $W \leftarrow V(G)$ 
for all configuration  $(v_r', R)$  do
    if  $|\text{Opt}(v_r', R, \text{NIL})| \leq |W|$  then
         $W \leftarrow \text{Opt}(v_r', R, \text{NIL})$ 
    end if
end for
return  $W$ 

```

4 Conclusions and Open Problems

We have proved that METRIC DIMENSION is NP-hard for planar graphs, even when the graph has bounded degree. We also gave a polynomial-time algorithm to solve the problem on outerplanar graphs. Our algorithm is based on an innovative use of dynamic programming which allows us to deal with *global problems* (in the sense described in the introduction), and which may be amendable to other such problems. One possible candidate is the *identifying codes problem*. Let $N_r(v) = \{u \in V(G) \mid d(u, v) \leq r\}$. Then given as input a graph G and an integer $r > 0$, the identifying codes problem is to find a smallest $D \subseteq V(G)$ such that $N_r(v) \cap D \neq \emptyset$ for all $v \in V(G)$, and $N_r(v) \cap D \neq N_r(u) \cap D$ for all distinct $u, v \in V(G)$. This problem is known to be NP-hard for general and planar graphs, and to have polynomial-time algorithms for cycles, paths, and trees (see [1] and references therein). We believe that the techniques developed in the present paper can yield a polynomial-time algorithm for the identifying codes problem on outerplanar graphs when r is part of the input.

We also pose some open problems for METRIC DIMENSION. First, it would be nice to extend our results to k -outerplanar graphs. Unfortunately, the technique described in the present paper does not seem to generalize in this direction, as edges and cut vertices are no longer convex separators in this case. Even if the problem turns out to be solvable on k -outerplanar graphs by a polynomial-time algorithm, it is not clear that such an algorithm could be used to derive a polynomial-time approximation scheme for PLANAR METRIC DIMENSION. The

quest for such an approximation scheme or even for a constant approximation algorithm is an interesting challenge in its own right.

Another interesting line of research is the *parameterized complexity* of METRIC DIMENSION. Daniel Lokshtanov [17] posed this problem at a recent Dagstuhl seminar on parametrized complexity. Moreover, he conjectured that the problem could be $W[1]$ -complete. We hope that our results can serve as a starting point for further study in this direction.

Acknowledgment: The authors thank David Johnson for sending a copy of the NP-completeness proof.

References

1. D. Auger, I. Charon, O. Hudry and A. Lobstein, Complexity results for identifying codes in planar graphs. *Intern. Trans. Operational Research* 17, 691–710, 2010.
2. R.F. Bailey, P.J. Cameron Base size, metric dimension and other invariants of groups and graphs. *Bulletin London Math. Society* 43 (2): 209–242, 2011.
3. Z. Beerliova, T. Eberhard, T. Erlebach, A. Hall, M. Hoffmann, M. Mihalak, L. Ram, Network Discovery and Verification. *IEEE J. Selected Areas in Communication* 24, 2168–2181, 2006.
4. B. Baker, Approximation algorithms for NP-complete problems on planar graphs. *JACM* 41, 153–180, 1994.
5. H. L. Bodlaender, Classes of Graphs with Bounded Treewidth. *Bulletin of the EATCS* 36, 116–125, 1988.
6. B. Courcelle, Graph rewriting: An algebraic and logic approach. *Handbook of Theoretical Computer Science*, vol. B, 194–242, Elsevier Science, 1990.
7. G. Chartrand, L. Eroh, M.A. Johnson, O.R. Oelmann, Resolvability in graphs and the metric dimension of a graph. *Discrete Applied Math.* 105, 99–113, 2000.
8. E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, M. Yannakakis, The Complexity of Multiterminal Cuts. *SIAM J. Computing* 23, 864–894, 1994.
9. R. Diestel, *Graph Theory*. Springer-Verlag, 2000.
10. D. Eppstein, Diameter and treewidth in minor-closed graph families. *Algorithmica* 27, 275–191, 2000.
11. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.
12. F. Harary, R.A. Melter, The metric dimension of a graph. *Ars Combinatoria* 2, 191–195, 1976.
13. M. Hauptmann, R. Schmied, C. Viehmann, On approximation complexity of metric dimension problem. *J. Discrete Algorithms*, in press, 2011.
14. J. Hopcroft, R.E. Tarjan, Efficient planarity testing. *JACM* 21, 549–568, 1974.
15. D.S. Johnson, personal communication.
16. S. Khuller, B. Raghavachari, A. Rosenfeld, Landmarks in Graphs. *Discrete Applied Math.* 70, 217–229, 1996.
17. D. Lokshtanov, Metric Dimension. In: Open Problems from Dagstuhl Seminar 09511, E. D. Demaine, M. T. Hajiaghayi, D. Marx, editors, 2009. Available at <http://erikdemaine.org/papers/DagstuhlFPT2009Open/paper.pdf>
18. P. Slater, Leaves of trees. *Congressus Numerantium* 14, 549–559, 1975.